

Laboratorio Nro. X: Escribir el Tema del Laboratorio

Nombre completo de integrante 1

Universidad Eafit
Medellín, Colombia
correointegrante1@eafit.edu.co

Nombre completo de integrante 2

Universidad Eafit
Medellín, Colombia
Correointegrante2@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

1. Este punto se logra utilizando las dos estructuras de datos propuestas para el problema. La primera, listas de adyacencias, funciona creando una lista de una lista de pares. La lista que contiene las otras listas se podría ver también como la lista de los vértices del grafo, donde cada sub-lista son los sucesores del vértice definido en la primera lista. La sub-listas son de pares ya que hay que almacenar el peso de cada arco hacia un vértice, guardando como el par entonces el número del vértice destino y su peso. Como algoritmos se utilizaron addArc donde se añade un nuevo par a un vértice (una de las sub-listas de la lista de adyacencia), getSucceros que devuelve todos los sucesores de un vértice recorriendo toda una sublista y devolviendo cada valor de los vértices de los pares que encuentre en dicha sub-lista, y por último getWeight que devuelve el peso entre dos vértices tomando dicho valor almacenado en un par. También se realizó el mismo ejercicio con matrices de adyacencia. La matriz consiste en una tabla donde las filas son el origen, las columnas el destino, y su valor, el peso que tiene dicha conexión. Para los algoritmos, addArc simplemente añade el valor del peso entre el vértice origen y el vértice destinación a la tabla, getSuccesors devuelve todos los vértices que tengan peso en una fila (vértice seleccionado) de la matriz, y getWeight devuelve el valor almacenado entre dos vértices en la matriz.
2. No hay una respuesta concreta para esta pregunta ya que ambas formas de representar un grafo tienen sus ventajas y desventajas. Las listas de adyacencias son muy útiles para representar grafos debilmente conexos, donde el número de vértices es mucho mayor que el promedio de sucesores que tenga un vértice, permitiendo ahorrar una gran cantidad de memoria ya que solo se almacena información esencial, mientras que usar las matrices de adyacencias podría dar ciertas ventajas representando grafos fuertemente conexos, ya que aunque gasta más memoria, esto le permite una implementación de algoritmos, por lo general, más eficiente que los que se logran con las listas de adyacencia.
3. En este caso es mejor utilizar una lista de adyacencias, ya que se está manejando una gran cantidad de información. Si se utiliza una matriz de adyacencia se ocuparía una cantidad de n^2 espacios de memoria, donde n es la cantidad de datos.
4. La mejor estructura de datos para representar este problema serían las listas de adyacencias. Esto es dado a que si los usuarios son vistos como vértices de un grafo, cada nodo casi no tiene conexiones a comparación del número total de vértices y por lo tanto es mucho mejor hacerlo con listas de adyacencia para ahorrar mucha memoria. Si son 100 millones de usuarios, hacerlo con matrices de adyacencias sería igual a guardar 1×10^{16} (100 millones por 100 millones) datos mientras que con las listas de adyacencia sería 2×10^{10} (100 millones por 200) que es mucho menor.

DOCENTE MAURICIO TORO BERMÚDEZ

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: mtorobe@eafit.edu.co

5. Para este problema la mejor opción sería usar una matriz de adyacencia ya que el problema podría ser representado como un grafo fuertemente conexo. Ya que mayoría, sino todas, las conexiones de este problema van a tener cierto valor diferente al nulo la mejor opción sería manejarlo con una matriz de adyacencia. La matriz permitiría una eficiencia mayor que las listas en este tipo de problema, ya que su estructura permite la implementación de algoritmos, como uno de búsqueda, más eficiente que el de las listas.
6. La estructura de datos que se escogió utilizar para este problema consta de una matriz de adyacencia, donde se tienen todos los nodos y arcos. Además, se tiene un arreglo donde se guardan los colores que tiene cada vértice. Primero se le asigna un color al primer vértice y se añade a una cola, en la cual se revisa cada uno de sus vértices adyacentes y se pintan con un color distinto al del que se encuentra en la cola y se remueve el inicial. Luego se añade el vértice que sigue a la cola y se sigue revisando que sus vecinos no estén coloreados, así hasta recorrer toda la cola. En caso de que suceda que dos vértices estén unidos y tengan el mismo color se imprime que el grafo “No es bicolorable”. Si se recorre toda la cola y se asignan colores a cada valor, se imprime que es “Bicolorable”.

4) Simulacro de Parcial

	0	1	2	3	4	5	6	7
0				1	1			
1	1		1			1		
2		1			1		1	
3								1
4			1					
5								
6			1					
7								

- 1.
- 2.


0->3,4
1->0,2,5
2->1,4,6
3->7
4->2
5->
6->2
7->

3. $O(n^2)$

5) Lectura recomendada (opcional)

- a) Título
- b) Ideas principales
- c) Mapa de Conceptos

6) Trabajo en Equipo y Progreso Gradual (Opcional)

	<p style="text-align: center;"> UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS </p>	<p>Código: ST245</p> <p>Estructura de Datos 1</p>
---	--	---

- a) Actas de reunión
- b) El reporte de cambios en el código
- c) El reporte de cambios del informe de laboratorio